

APPLICATION FOR UNITED STATES PATENT
SYSTEM AND METHOD FOR CONFIGURING
CONTENTS OF NETWORK MANAGEMENT
NOTIFICATIONS

By Inventors:

Keith McCloghrie
170 W. Tasman Drive
San Jose, CA 95134-1706

Ramanathan Kavasseri
170 W. Tasman Drive
San Jose, CA 95134-1706

Sandra Durham
170 W. Tasman Drive
San Jose, CA 95134-1706

Assignee: Cisco Technology
(A California Corporation)
170 W. Tasman Drive
San Jose, CA 95134-1706

Entity: Large

RITTER, VAN PELT & YI LLP
4906 El Camino Real, Suite 205
Los Altos, CA 94022
(650) 903-3500

SYSTEM AND METHOD FOR CONFIGURING CONTENTS OF NETWORK MANAGEMENT NOTIFICATIONS

BACKGROUND OF THE INVENTION

The present invention relates generally to network management, and more particularly, to configuring the contents of an SNMP notification.

It is desirable to manage and control nodes of a network such as a TCP/IP network from one or more network management stations connected to the network. The network management stations monitor and control other nodes in the network including hosts, routers, etc. A protocol known as the Simple Network Management Protocol (SNMP) is used to communicate management information between network management stations (NMS) and SNMP agent software operating on other network nodes, or the same node as the network management station. SNMP is described in Case, RFC 1157, "A Simple Network Management Protocol (SNMP)," (Internet Engineering Task Force May 1990), the contents of which are incorporated by reference herein. Using SNMP in a TCP/IP network, a network management station may monitor traffic at a remote node and modify network operation parameters at the remote node.

Fig. 1 is a block diagram illustrating an exemplary network utilizing SNMP. The network includes at least one management station 10 which controls a plurality of nodes such as workstation 11, bridge 12, router 14, printer 15, or other peripheral devices (e.g., personal computers, hubs, repeaters). The management station 10 includes a SNMP manager 16 (command generator and notification receiver). In order for an SNMP management station 10 to manage a node, the node must be able to run an SNMP agent 17 (command responder and notification originator), which is an SNMP management process typically encoded in software (some nodes may use a hardware implementation).

To facilitate SNMP operation, nodes of the network including the network management stations 10 maintain network management information databases known as MIBs (management information bases) 18. MIBs are described in McCloghrie, et al., RFC 1213, "Management Information Base for Network Management of TCP/IP Based Internets: MIB-II" (Internet Engineering Task Force March 1991), the contents of which are incorporated by reference herein.

The MIB 18 at each node consists of many individual objects, each having one or more instance. Each instance has a single value. For example, a particular node may include within its MIB an object whose value specifies the total number of IP datagrams received at that node, including those received in error. Each MIB 18 includes a large number of such objects. The object instances and their corresponding values are carried

in packets called protocol data units (PDUs) and contain operating parameters, statistics, and control information for the element and its components.

Each MIB object is uniquely identified by a series of integers called an Object Identifier (OID). For example, the object ifInOctets is represented by the OID 1.3.6.1.2.2.1.10. There can be many instances of an object in existence on an agent. For example, there is one instance (and value) of ifInOctets for every physical interface known to the agent. Each instance is identified by appending one or more additional integers to the OID (e.g., 1.3.6.1.2.2.1.10.x, where x is an integer greater than zero). The first element of this specific OID identifies an overall object identifier domain allocated to the ISO. The second element has the value 3 which is defined as indicating allocation toward an organization under the auspices of the ISO. The third element identifies the US Department of Defense (DOD) as the responsible organization. The fourth element has the value of 1 and identifies the Internet. The fifth element indicates that the identifier is used for management. The remaining elements identify the particular object types with greater specificity.

The organization of objects within an MIB may be illustrated as a tree structure. An example of a portion of such a tree structure is illustrated in Fig. 2. The leaves of the tree instances of individual objects. Fig. 2 shows the MIB tree from its root, "ISO", to some of its lower branches. A first branch 21 of primary interest includes the standard MIB objects defined by RFC 1213. A second branch 23 includes MIB objects defined

for use by Cisco Systems. The illustration of Fig. 2 does not extend down to individual leaves but rather indicates the prefix of elements in OIDs for various classes of objects. For example, OIDs from MIB objects pertaining to TCP would begin with 1.3.6.1.2.1.6.

5 The management station communicates with the agents over the network using the SNMP protocol, which allows the management station to query the state of the agent's local objects and modify them if necessary. SNMP is a request-response protocol by which the variables of an agent's MIB may be inspected or altered. The protocol is described in RFC 1905, "Protocol Operations for Version 2 of the Simple Network Management Protocol", Case, McCloghrie, Rose & Waldbüßer, January 1996, the
10 contents of which are incorporated by reference herein.

A typical SNMP operation involves management station sending an SNMP message to agent requesting a particular local parameter. The agent then recovers this parameter by accessing a particular object in the MIB and responds to the request with a reply to the management station including the parameter value. SNMP operations are
15 defined on a list of MIB variable instances called a variable binding (varbind) list. Each element of a varbind list describes a specific MIB variable instance. A varbind element specifies three MIB instance attributes: its object identifier, data type, and value.

The management protocol provides for the exchange of messages which convey management information between the agents and management stations. For example, the
20 management station may send a request to an agent asking it for information or

commanding it to update its state in a certain way. The agent typically replies with the requested information or confirms that it has updated its state as requested. Conventional message types include Get, GetNext, GetBulk, Set, Trap, and Inform. The Get operation is used by the system to retrieve the value of one or more object instances from an agent.

5 The GetNext operation is used by the system to retrieve the value of the next object instance in a table or list within an agent. The GetBulk operation is used by the system to efficiently retrieve large blocks of data, such as large tables. The Set operation is used by the network management station to set the values of object instances within an agent.

10 The Trap operation is used by agents to asynchronously inform the network management station of a event of interest to one or more network management stations. Notification originators can generate informs, which are notifications that should be acknowledged by notification receivers. In an acknowledgement is not received within a configurable time-window, the notification originator attempts to resend the inform up to a maximum of N returns, where N is a configurable integer. The Trap and Inform operations may be
15 used to send notifications to the manager.

Notifications are unsolicited messages sent from an agent to the SNMP manager to apprise the manager of network events. SNMP notifications allow the agent to initiate communication with management applications when an event of interest to one or more network management stations takes place. These events include cold start, warm start,
20 link down, link up, authentication failure, neighbor loss, and enterprise. SNMP

notifications contain a sequence of SNMP objects that typically provide context information on why each notification was generated.

Many networks contain components manufactured by several different companies. In order for the management station to communicate effectively with these varied devices, the information maintained by the agents must be rigidly specified. 5
SNMP therefore describes the exact information each agent 17 must maintain and the format in which it must be maintained in the MIB 18. At times, newer versions of MIBs result in new objects being added to notifications. Also, notifications are often generated with different variable bindings than those that should be listed in the MIB, either in error or because a network element has not kept up with newer versions of MIBs. 10
Furthermore, it is often desirable to add specific objects to a notification to provide additional flexibility in management of a network.

There is, therefore, a need for a system and method for instructing an agent which objects to send for each notification so that notifications can be easily reconfigured by a 15
management station.

SUMMARY OF THE INVENTION

A method and system for configuring contents of network management notifications are disclosed.

5 A method of the present invention includes receiving at a management station a list of notifications supported by an agent and sending a message to the agent specifying objects for each of the notifications. A notification containing the specified objects is received at the management station upon the occurrence of an event at the agent.

10 The messages may be sent using SNMP protocol. The list of notifications and objects may be transferred between the management station and agent in the form of a management information base. The notifications may be sent to the management station as a Trap or Inform message.

15 In another aspect of the invention, a method for sending SNMP notifications from an agent to a management station comprises receiving a message from the management station specifying objects for each notification supported by the agent. The agent then modifies a list of objects for each of the notifications and sends a notification containing the specified objects upon occurrence of an event.

A computer program product of the present invention comprises code that receives at a management station a list of notifications supported by an agent, code that sends a message to the agent specifying objects for each of the notifications, and code

that receives a notification containing the specified objects. The product further includes a computer-readable storage medium for storing the codes.

5 A system of the present invention includes a processor that receives information specifying contents of notifications supported by an agent within a network at a management station and sends instructions from the management station to the agent to modify the contents of the notification to a preferred configuration. The system further includes a storage medium having the preferred configuration stored thereon.

10 In another aspect of the invention, a system for configuring the contents of a network management station comprises means for receiving information specifying contents of notifications supported by an agent within a network at a management station and means for sending instructions from the management station to the agent to modify contents of the notifications.

15 The above is a brief description of some deficiencies in the prior art and advantages of the present invention. Other features, advantages, and embodiments of the invention will be apparent to those skilled in the art from the following description, drawings, and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram illustrating an exemplary network utilizing a network management protocol.

Fig. 2 is a schematic illustrating a portion of a tree structure used to specify MIB object identifiers.

Fig. 3 is a schematic illustrating an example of a computer system that can be utilized to execute software of an embodiment of the present invention.

Fig. 4 is a system block diagram of the computer system of Fig. 3.

Fig. 5 is a schematic of a router which may represent a network node according to the present invention.

Fig. 6 is a flowchart illustrating a process for defining objects for a notification.

Fig. 7 is an example of a notification table specifying the notifications an agent supports.

Fig. 8 is a notification variable binding table specifying the variable bindings the agent is configured to send for each of the notifications listed in the table of Fig. 7.

Fig. 9 is a portion of a modified notification variable binding table sent to the agent to instruct the agent how to configure notifications.

Corresponding reference characters indicate corresponding parts throughout the several views of the drawings.

DETAILED DESCRIPTION OF THE INVENTION

5 The following description is presented to enable one of ordinary skill in the art to make and use the invention. Descriptions of specific embodiments and applications are provided only as examples and various modifications will be readily apparent to those skilled in the art. The general principles described herein may be applied to other embodiments and applications without departing from the scope of the invention. Thus, the present invention is not to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features described herein. For purpose of clarity, details relating to technical material that is known in the technical fields related to the invention have not been described in detail.

10 The present invention operates in the context of a data communication network including one or more multiple nodes. Nodes of a network may be user workstations, servers, routers, etc. Certain nodes of the network may be network management stations (NMS) 10 used to monitor and control overall network operation (Fig. 1). As described below, the method and system of the present invention allow a management station 10 to dynamically add new objects to a notification and reorder the objects in the notification so that the management station receives specific information from an agent 17 upon the occurrence of an event. This allows system administrators to tailor notifications to meet specific needs of a network. The present invention further provides a method for

determining the objects contained within a device's notification before a notification is received by the management station 10.

Fig. 3 illustrates an example of a computer system that may be used to execute software of an embodiment of the invention. The computer system 20 includes a display 22, screen 24, cabinet 26, keyboard 28, and mouse 30 which may include one or more buttons for interacting with a GUI (Graphical User Interface). Cabinet 26 houses a CD-ROM drive 32, system memory 42 and a hard drive 44 (see Fig. 4) which can be utilized to store and retrieve software programs incorporating computer code that implements aspects of the invention, data for use with the invention, and the like. Although CD-ROM 34 and floppy disk 35 are shown as exemplary computer readable storage media, other computer readable storage media including tape, flash memory, system memory, and hard drive may be utilized. Additionally, a data signal embodied in a carrier wave (e.g., in a network including the Internet) may be the computer readable storage medium.

Fig. 4 shows a system block diagram of computer system 20 used to execute software of an embodiment of the invention. Computer system 20 further includes subsystems such as a central processor 40, system memory 42, fixed storage 44 (e.g., hard drive), removable storage 46 (e.g., CD-ROM drive), display adapter 48, sound card 50, transducers 52 (speakers, microphones, and the like), network interface 54, and printer/fax/scanner interface 56. Other computer systems suitable for use with the invention may include additional or fewer subsystems. For example, computer system 20

may include more than one processor 40 (i.e., a multi-processor system) or a cache memory.

The system bus architecture of computer system 20 is represented by arrows 58 in Fig. 4. However, these arrows are only illustrative of one possible interconnection scheme serving to link the subsystems. For example, a local bus may be utilized to connect the central processor 40 to the system memory 42 and display adapter 48. Computer system 20 shown in Figs. 3 and 4 is only one example of a computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems may also be utilized.

Some of the nodes in a network that employs the present invention may be network devices such as routers and switches. For example, some of the nodes may be specially configured routers such as those available from Cisco Systems, Inc. of San Jose, California. A general architecture for some of these machines will appear from the description given below. In an alternative embodiment, a router or switch may be implemented on a general purpose network host machine such as the computer system of Figs. 3 and 4.

Referring now to Fig. 5, a router 60 suitable for implementing the present invention includes a master central processing unit (CPU) 61, interfaces 68, and a bus 65 (e.g., a PCI bus). As shown, CPU 61 includes a memory 62 and a processor 63. When acting under the control of appropriate software or firmware, CPU 61 is responsible for

such router tasks as routing table computations, network management, and general processing of packets. It preferably accomplishes all these functions under the control of software including an operating system and any appropriate applications software. CPU 61 may include one or more processors 63 such as a processor from the Motorola family or microprocessors of the MIPS family of microprocessors. In an alternative embodiment, processor 63 is specially designed hardware for controlling the operations of router 60. Memory 62 can be non-volatile RAM and/or ROM. However, there are many different ways in which memory could be coupled to the system.

The interfaces 68 are typically provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receiving of data packets over the network and sometimes support other peripherals used with the router 60. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, serial interfaces, and the like. In addition, various high-speed interfaces may be provided such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces and the like. Generally, these interfaces may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile RAM. The independent processor may control such communications intensive tasks as packet switching, media control, and management. By providing separate processors for the communications intensive tasks, these interfaces allow the master microprocessor 61 to efficiently perform routing

computations, network diagnostics, security functions, etc. Router 60 may further include a packet memory 72 for intermediate storage of packets forwarded by the router.

Although the system shown in Fig. 5 is one specific router usable in connection with the present invention, it is by no means the only architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations is often used. Further, other types of interfaces and media may also be used with the router.

Regardless of a network device's configuration, it may employ one or more memories or memory modules (including memory 62) configured to store program instructions for the network management operations described herein. The program instructions may control the operation of an operating system or one or more applications, for example. The program instructions and operating system may be stored on any of the media discussed in connection with the computer system of Figs. 3 and 4, for example.

In one embodiment, the present invention operates in conjunction with SNMP which is described in Case, et al., "A Simple Network Management Protocol (SNMP)," RFC 1157, (IETF May 1990), the contents of which are incorporated by reference herein. As shown in Fig. 1, a network control entity (e.g., SNMP manager) 16 resides on a network management station 10. Agent entity 17 resides on other nodes 11, 12, 14, 15 of the network managed by the manager 16. Manager 16 and agents 17 interact according

to a protocol defined by SNMP. The SNMP manager 16 may set system parameters, e.g., TCP/IP parameters at the node occupied by agent 17.

The SNMP manager 16 may include one or more command generators and notification receivers. The manager 16 may further comprise a dispatcher which includes a PDU dispatcher, and transport mapping system, along with a message processing subsystem, and a security subsystem. The manager 16 is operable to query agents 17, get responses from agents, set variables in agents, and acknowledge asynchronous events from agents. The agent 17 may include a dispatcher, message processing subsystem, and a security subsystem along with a command responder application, access control, notification originator application, and proxy forwarder application. The notification originator application generates SNMP messages containing Notification-Class PDUs (e.g., Trap PDU or Inform PDU). The command responder and notification originator applications are in communication with MIB instrumentation. The agent 17 stores and retrieves management data as defined by the Management Information Base (MIB) 18 and can asynchronously signal an event to the management station 10. It is to be understood that the management station 10 and nodes 11, 12, 14, 15 shown in Fig. 1 and described herein are only one example of a network and that the manager and agents may have configurations other than shown and described herein, without departing from the scope of the invention. Furthermore, management protocols other than SNMP may be used.

As previously described, the Management Information Base (MIB) 18 is accessible to SNMP manager 16 to facilitate managing network information. MIB 18 includes object identifiers (OIDs) that identify objects storing particular pieces of information about network operation. Each object includes the OID and a object value which is indicative of network operation at a particular node. An MIB 18 directly accessible to agent 17 stores MIB objects for the node of the agent. Each object includes the OID and the object value. The MIB 18 is a collection of definitions, which define the properties of the managed object within the device to be managed. MIB contents are described in McCloghrie, et al., "Management Information Base for Network Management for TCP/IP-Based Internets: MIB-II", (IETF March 1991), the contents of which are incorporated by reference herein. Examples of MIB objects include things such as: the number of outbound packets that could not be transmitted due to errors, the length of the output packet queue, text string indicating the physical location of the node, etc.

Typically, there are many instances of each managed object type within a management domain. The method for identifying instances specified by the MIB module does not allow each instance to be distinguished amongst the set of all instances within a management domain. Instead it allows each instance to be identified only within some scope of context, where there are multiple such contexts within the management domain. Often a context is a physical device, or a logical device, although a context can also encompass multiple devices, or a subset of a single device, or even a subset of multiple

devices, but a context is always defined as a subset of a single SNMP entity. Thus, in order to identify an individual item of management information within the management domain, its contextName and contextEngineID must be identified in addition to its object type and its instance. For example, the managed object type ifDescr is defined as the description of a network interface. To identify the description of a device's first network interface, five pieces of information are needed, the snmpEngineID of the SNMP entity which provides access to the management information at the device, the contextName, the contextEngine ID, the managed object type (e.g., ifDescr), and the instance. Since the notification contains objects that are specified in the MIB definition, the instances of the object will be borrowed when the objects within the notification are modified, as described below.

The management station 10 interacts with the agents 17 using the SNMP protocol. This protocol allows the management station to query the state of an agent's local objects, and change them if necessary. Most of SNMP consists of this query-response type communication. However, sometimes events happen that are not planned. For example managed nodes can crash and reboot, interfaces can go down and come back up, or congestion can occur. When an agent 17 notices that an event of interest to an NMS has occurred, it immediately reports the event to all management stations 10 in its configuration list. This report is called an SNMP notification. The report typically states that some event has occurred. It is up to the management station 10 to then issue queries to find out all the details.

When an application decides to send a notification, it may call one of the following functions, for example:

`snmp:send_notification(Agent, Notification, Receiver [NotifyName, Varbinds, ContextName])`

5 The agent 17 may then perform a get operation to retrieve the object values which are defined in the notification specification. The notification is sent to all managers 16 defined in the target and notify tables, either unacknowledged as Traps, or acknowledged as Inform requests.

10 Fig. 6 is a flowchart illustrating a process of the present invention for specifying objects to be contained within the notification. The method is preferably performed by software, however, the method may also be implemented in hardware. The method is preferably performed by the management station 10, however, it may be performed by one of the agents 17, or any other device in the network.

15 At step 100, the network management station 10 sends a Get or Getnext request to all of the agents 17 asking for a list of notifications that it supports. The request may also ask for the associated variable bindings (varbinds) (i.e., OID + value), or a separate request may be sent. The agents 17 then respond to the management station 10 with a list of notifications it supports (step 102). The agent 17 may send a table as shown in Fig. 7, listing the notifications that it currently supports along with a table, shown in Fig. 8,

listing the notification varbinds. The first table may include, for example, link up, link down, ciscoPingCompletion, or any other event that results in a notification being sent from the agent 17, such as authentication failure, cold start, warm start, neighbor loss, (Fig. 7). It is to be understood that notifications may be supported for events other than those listed herein. The second table includes a list of varbinds for each notification supported by the agent (Fig. 8). Each varbind includes an object identifier (OID) and the corresponding value (as described above). The management station 10 next tells each agent 17 which objects to include in each of the notifications and the order of the objects (step 104). The management station 10 may provide this information as a MIB containing a modified list of varbinds, as shown in Fig. 9; for example. Fig. 9 shows a portion of a table containing a modified varbind list and the corresponding notification index and varbind index. The agent's notifications originally contained the following varbinds: ifIndex; ifDescr; ifType; ciscoPingCompleted; ciscoPingSentPackets; and ciscoPingReceivedPackets. The management station added the following varbinds: ifAdminStatus and ifOperStatus. It also changed the location of the original three varbinds to insert the two new varbinds. The new list of varbinds may include additional varbinds or change the context of the existing varbinds. When an event occurs, the agent 17 sends to the management station 10 a notification containing the varbinds requested by the management station (steps 106 and 108).

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there

could be variations made to the embodiments without departing from the scope of the present invention. Accordingly, it is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

5